Gene Trees: Phylogenetic Trees from Amino Acid Sequences

Beckie Chan

CRPC-TR99810-S
August 1999

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

Submitted November 1999

Gene Trees: Phylogenetic Trees from Amino Acid Sequences

Beckie Chan
pchan543@cs.caltech.edu
CRPC Summer Research Program in Parallel
Computing for Undergraduate Women
Department of Computer Science 256-80
California Institute of Technology
Pasadena, CA 91125
August 20, 1999

Introduction

The goal of this program is to classify genes into families based on their nucleotide and amino acid sequences. From one gene, the program will find related genes, sort them into subfamilies, families, and super families, and organize them into phylogenetic trees. Svjetlana Miocinovic, an undergraduate at the California Institute of Technology, and I worked with biologists from the University of California, Irvine, Dr. George A. Gutman and Dr. K. George Chandy. To maximize efficiency, this program has been broken up into two parts. Svjetlana used similarity search programs to find related genes given one gene sequence. My part involved taking these related gene sequences to create a phylogenetic tree. The whole program has been written in Java, so it can easily be made web accessible in the future. Once this website has been created, biologists, doctors, and many others can take advantage of it.

The current process used to generate sequence alignments and trees can be extremely long and tedious. One begins by obtaining sequence files from a genetic sequence database (e.g. NCBI's GenBank [14]). Then, coding region nucleotide sequence files and amino acid sequence files are created. These sequences are then formatted into another file for alignment. Once the sequences are aligned, the output is converted into a NEXUS format. This NEXUS file must be manually edited to include only the regions where significant alignment exists among the sequences. Finally, after this has been done, a phylogenetic tree program is used to create the tree.

To make this process less laborious, a Java application is now available to go directly from the given sequences to the phylogenetic tree, without the need for manual tasks. The program is composed of four classes: Alignment, GroupMatch, MakeTree, and MainTree. This document explains the process of going from amino acid sequence to phylogenetic tree.

Clustal W 1.7: Multiple Sequence Aligning

Clustal W is a multiple sequence aligning program for DNA and proteins. Free versions of this program are readily available online for PC, UNIX, and Macintosh operating systems [3].

In order for a tree to be created, the amino acid sequences must first be aligned. Amino acid sequences are used because they contain more information than nucleotide sequences. Clustal W accepts an input file which contains all the sequences that need to be aligned. The sequences in this input file can be in various formats: NBRF/PIR, EMBL/SwissProt, and FASTA. My program writes the sequences in the FASTA format. This format is characterized by a ">" followed by the title of the gene and then the amino acid sequence, which must be on the next line after the title. Everything after the title is taken as one sequence until the next ">" is seen.

e.g.
```
>L02752
MTVATGDPADEAAALPGHPQDTYDPEADHECCERVVINISGLRFETQLKTLAQFPETLLGDPKKRMRYFD
PLRNEYFFDRNRPSFDAILYYYQSGGRLRRPVNVPLDIFSEEIRFYELGEEAMEMFREDEGYIKEEERPL
PENEFQRQVWLLFEYPESSGPARIIAIVSVMVILISIVSFCLETLPIFRDENEDMHGSGVTFHTYSNSTI
GYQQSTSFTDPFFIVETLCIIWFSFEFLVRFFACPSKAGFFTNIMNIIDIVAIIPYFITLGTELAEKPED
AQQGQQAMSLAILRVIRLVRVFRIFKLSRHSKGLQILGQTLKASMRELGLLIFFLFIGVILFSSAVYFAE
ADERESQFPSIPDAFWWAVVSMTTVGYGDMVPTTIGGKIVGSLCAIAGVLTIALPVPVIVSNFNYFYHRE
TEGEEQAQYLQVTSCPKIPSSPDLKKSRSASTISKSDYMEIQEGVNNSNEDFREENLKTANCTLANTNYV
NITKMLTDV
```

The number of characters on each line is insignificant, and the amino acid sequence can be written in upper or lower case letters. Furthermore, with the FASTA format, the user does not need to indicate whether the sequences are proteins or DNA. Clustal W will read the number of A, C, G, T, and U characters. If this amount is greater than 85% of the total sequence, the sequences will be considered DNA. Otherwise, they will be considered proteins. Also, in the 1.7 version of Clustal W,

dynamic memory allocation is used, so there is no limit to the length or number of sequences.

Once all of the sequences have been entered into a .clu file, Clustal W is called to run a full multiple alignment. Clustal W works with menu options or command line interface. In my program, the callClustal method in the Alignment class calls Clustal W with the command line "clustalw filename.clu," where filename is the name entered by the user. This tells Clustal W to do a complete multiple alignment on "filename.clu." First, a dendrogram is created, "filename.dnd." This rough phylogenetic tree tells Clustal W the approximate relationships between the sequences. Using the information in the dendrogram, a "filename.aln" file can be generated. This file contains the aligned sequences along with a conservation line. The conservation line is made up of spaces and three symbols: asterisks, periods, and colons. The asterisk indicates columns where the residue is fully conserved. A colon is placed when residues with similar physicochemical properties are fully conserved. These "strong" groups include STA, NEQK, NHQK, NDEQ, QHRK, MILV, MILF, HY, and FYW. A period indicates the presence of a "weaker" group that is fully conserved. Some examples of "weaker" groups are CSA, ATV, SAG, STNK, STPA, SGND, SNDEQK, NDEQHK, NEQHRK, FVLIM, and HFY.

The following is an example of an .aln file created by Clustal W:

```
L02752 --------------------MTVATGDPADEAAALPGHPQDTYDP-----EADHECCERV
M81352 ------------------------------------------------------------
L02750 --------------------MTVMSGENVDEASAAPGHPQDGSYPR-QADHDDHECCERV
M55514 ELSEEEEDEEEEEEEEEEEGRFYYSEDDHGDECSYTDLLPQDEGGGG-YSSVRYSDCCERV
X16002 ELSEEEEDEEEEEEEEEEEGRFYYSEEDHGDGCSYTDLLPQDDGGGGGYSSVRYSDCCERV

L02752 VINISGLRFETQLKTLAQFPETLLGDPKKRMRYFDPLRNEYFFDRNRPSFDAILYYYQSG
M81352 ---------------------------------------------NRPSFDGILYYYQSG
L02750 VINISGLRFETQLKTLAQFPNTLLGNPKKRMRYFDPLRNEYFFDRNRPSFDAILYYYQSG
M55514 VINVSGLRFETQMKTLAQFPETLLGDPEKRTQYFDPLRNEYFFDRNRPSFDAILYYYQSG
X16002 VINVSGLRFETQMKTLAQFPETLLGDPEKRTQYFDPLRNEYFFDRNRPSFDAILYYYQSG
                                                    ******.********
```

Homologous Alignment

After the .aln file is created, specific regions must be isolated where there may be homologous alignments. Sequences are homologous if they have a common evolutionary origin. I began isolating similar sections by observing the conservation line of the .aln file. However, this method proved inefficient because symbols will not be placed if there is a gap in the column of residues. For example, even if all the residues in the column match but one gap exists, an asterisk will not be placed.

Hence, before grouping sections, the conservation line must be modified. This was done in the putSymbols method of the Alignment class. The residues in each column are analyzed for a high percentage of similarity. If the same residue appears in more than 50% of the total column (e.g. D/-/E/D/D), an exclamation point is placed in the conservation line. Also, if the sum of the two most common residues is greater than 75% of the total, an exclamation point is placed. For instance, if a column contains the residues M/-/M/F/F, an exclamation point will be placed. Although M's make up only 40% of the sequence and F's makes up only 40% of the sequence, the column still has a strong indication of homology because the sum of these two most common residues is 80%. After modification, all of the asterisks will be replaced by an exclamation point and most of the periods and colons will also be replaced. The .aln file will look like this:

```
L02752 --------------------MTVATGDPADEAAALPGHPQDTYDP-----EADHECCERV
M81352 ------------------------------------------------------------
L02750 --------------------MTVMSGENVDEASAAPGHPQDGSYPR-QADHDDHECCERV
M55514 ELSEEEEDEEEEEEEEEEEGRFYYSEDDHGDECSYTDLLPQDEGGGG-YSSVRYSDCCERV
X16002 ELSEEEEDEEEEEEEEEEEGRFYYSEEDHGDGCSYTDLLPQDDGGGGGYSSVRYSDCCERV
                           !!! ! !!!!! !!!!!! ! !!!!!!!!

L02752 VINISGLRFETQLKTLAQFPETLLGDPKKRMRYFDPLRNEYFFDRNRPSFDAILYYYQSG
M81352 ---------------------------------------------NRPSFDGILYYYQSG
L02750 VINISGLRFETQLKTLAQFPNTLLGNPKKRMRYFDPLRNEYFFDRNRPSFDAILYYYQSG
M55514 VINVSGLRFETQMKTLAQFPETLLGDPEKRTQYFDPLRNEYFFDRNRPSFDAILYYYQSG
X16002 VINVSGLRFETQMKTLAQFPETLLGDPEKRTQYFDPLRNEYFFDRNRPSFDAILYYYQSG
       !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

One can see the drastic difference in the conservation line of this new modified .aln file and its original. This new format indicates all the areas of good alignment. If isolating the good sections were solely based on the original conservation line, many residues would be missed.

Once all of the exclamation points have been placed, a .ext file is created. This file contains all the information of the .aln

file, except in a different format. The .aln file breaks the sequences into 60 characters for each line. This makes reading the sequences difficult because they are not continuous. Every time the first 60 characters are read, the cursor must skip past the other sequences and skip the gene name to continue reading the sequence. To make reading the sequences less complicated, the putTogether method of the GroupMatch class creates a .ext file which places each sequence on one line only. The conservation line is also included.

Now, sections, which show homology, can be isolated. Each sequence is read into an array of strings. The conservation line is put into its own separate string. The first occurrence of an exclamation point is found. From this position, the cursor reads ten more symbols to the right, incrementing a counter each time it reads an exclamation point. If this group of ten is made up of over 30% exclamation points, then the program keeps working on this region; otherwise it moves on to find another group of ten. When a 'good' group is found, a variable called startkeep is set to the beginning of this section of ten and stopkeep is set to the end of this section. Then, the cursor is moved to the left of startkeep to check for close by exclamation points. If a ?!? is read while searching to the left of startkeep, startkeep will be reset to that new index. Once the cursor reads six consecutive spaces, the program stops searching backwards. The same procedure is done to stopkeep except the cursor reads more symbols to the right. If a ?!? is found, the value of stopkeep is updated to that position. This process ends once six consecutive spaces are read. When startkeep and stopkeep values are defined, a substring is created with those boundaries. The total number of exclamation points in this substring is counted. If the substring is short (made up of less than 16 residues), exclamation points must make up over 75% of the string for the section to be kept. If the substring is average (16-27 residues), exclamation points must make up over 50% of the string. If the substring is long (over 27 residues), exclamation points must make up at least 37% of the string. These variables can be changed to acquire the best results. Currently there are three different choices. When calling the program, the user can enter 1, 2, or 3. Choice 1 is described by the parameters above. Choice 2 has the same requirements for a short region. However, for an average or long region, 37% of the string must have exclamation points. Choice 3 also has the same requirements for a short region. But, for an average or long region to be conserved, the substring must have at least 60% of !'s. These conserved sections are placed in a .tru file. After the substring is kept or ignored, the process moves on to the next ?!? until the end of the sequences.

Eventually, the criteria for isolating the sections will be read from a data file. This will allow the user to enter his own definitions for a long, average, short section. Also, the user can set his own percentage values.

The following are two of the sections conserved from the original .ext file. These two sections would have been completely missed if the original conservation line was used to isolate parts.

```
L02752 MTVATGDPADEAAALPGHPQDTYDP
M81352 -------------------------
L02750 MTVMSGENVDEASAAPGHPQDGSYP
M55514 FYYSEDDHGDECSYTDLLPQDEGGG
X16002 FYYSEEDHGDGCSYTDLLPQDDGGG

L02752 DHECCERVVINISGLRFETQLKTLAQFPETLLGDPKKRMRYFDPLRNEYFFDRNRPSFDAILYYYQSG
M81352 ------------------------------------------------------NRPSFDGILYYYQSG
L02750 DHECCERVVINISGLRFETQLKTLAQFPNTLLGNPKKRMRYFDPLRNEYFFDRNRPSFDAILYYYQSG
M55514 YSDCCERVVINVSGLRFETQMKTLAQFPETLLGDPEKRTQYFDPLRNEYFFDRNRPSFDAILYYYQSG
X16002 YSDCCERVVINVSGLRFETQMKTLAQFPETLLGDPEKRTQYFDPLRNEYFFDRNRPSFDAILYYYQSG
```

Phylogenetic Tree

Once the "good" regions of the sequences have been isolated, a phylogenetic tree can now be created. A phylogenetic tree displays many genes and their evolutionary relationships. The tree consists of nodes and branches. Every child node has a parent node, except for the root. The genes are placed at the leaf nodes. Genes, which are closely related to each other, are closely joined together in the tree. Genes, which are more distantly related, are farther apart in the tree.

Phylogenetic trees can be displayed in two ways: rooted and unrooted [11]. Unrooted trees have branches that can spread in any direction. Hence, the evolution of the tree is undefined. Rooted trees have a root node located on the left and have branches that extend to the right. If the location of the root is not known, trees are best left unrooted.

Various tree-making programs were considered for this part of the program. PAUP, Phylogenetic Analyses Using Parsimony, is a widely used program for generating evolutionary trees. Unfortunately, this program could not be used because the new 4.0 version of this program was not available for UNIX operating systems. Another tree program, which was considered, was NJBAFD, neighbor-joining tree construction from allele frequency data. However, this program was unsuitable for the program because it required the allele frequency data in order to create the tree.

Finally, Clustal W seemed to be the best choice. Not only does Clustal W align sequences, but it also creates phylogenetic trees. Clustal W uses the NJ, neighbor joining, method designed by Saitou and Nei [8]. The distances between all pairs of sequence are calculated first. Then, the NJ method is applied to the distance matrix.

The tree-generating program of Clustal W only accepts aligned sequences that are in theNBRF/PIR, EMBL/SwissProt, Clustal, or FASTA formats. The conserved regions in the .tru file are already in the Clustal format and have also already been aligned. Therefore, the program goes immediately to generating the tree.

Next, the MakeCTree method in the MakeTree class calls Clustal W to create an unrooted tree. The program outputs the tree in a .ph file. This output is in a Phylip format, also known as the New Hampshire format. It contains a group of parentheses with sequence names and branch lengths. The tree is not displayed visually. To view the tree, one must draw it out on paper or use one of the tree drawing programs in the PHYLIP package.

The following is a sample of the .ph file.

```
(
(
L02752:0.03745,
M81352:0.03837)
:0.04873,
L02750:0.10365,
(
M55514:0.00872,
X16002:0.01058)
:0.20533);
```

The PHYLIP package [4], also available on the web, contains three tree-displaying programs: RETREE, DRAWGRAM, and DRAWTREE. RETREE was the most user friendly. This program is completely driven by menus. The user can call RETREE and a menu of options will appear:

```
Tree Rearrangement, version 3.573c
Settings for this run:
U Initial tree (arbitrary, user, specify)? User tree from tree file
N Use the Nexus format to write out trees? No
0 Graphics type (IBM PC, VT52, ANSI)? ANSI
W Width of terminal screen, of plotting area? 80, 80
L Number of lines on screen? 24
Are these settings correct? (type Y or the letter for one to change)
```

After agreeing to the above specifications, the program looks for an input file called intree. If this file is not found, the program prompts the user for the name of an input file. Enter the .ph file created by Clustal W. The visual display of the tree will then appear on the screen.

```
                 ,----------1:L02752
  ,--------------7
  !              `-----------2:M81352
--6
  !  ,-------------------------------3:L02750
  `--8
     !                                                            ,--4:M55514
     `-----------------------------------------------------------9
                                                                  `--5:X16002
```

In this example, L02752 and M81352 are grouped together. This is consistent with expected results because L02752 and M81352 are both from the Kv1.2 family. M55514, human Kv1.4, is also correctly paired with X16002, rat Kv1.4. L02750 is on its own branch because it is in the Kv1.1 family.

Once the tree is drawn, the user has a number of options to manipulate the tree, including rerooting the tree. When exiting the program, the user can choose to save the tree in a file called ?outtree.? One has the option of saving the rooted tree or the unrooted tree. The file will be in the Newick tree format [5]. This format uses nested parentheses and has a semicolon to represent the end of the tree.

```
rooted:
((L02752:0.03745,M81352:0.03837):0.04873,(L02750:0.10365,(M55514:0.00872,
X16002:0.01058):0.20533):0.00000);

unrooted:
```

```
((L02752:0.03745,M81352:0.03837):0.04873,L02750:0.10365,(M55514:0.00872,
X16002:0.01058):0.20533);
```

For the rooted tree, each pair of parentheses represents an interior node. Inside these parentheses are representations of nodes that are immediate descendents from that original node. In the above example, branch lengths are also included. For the unrooted tree [6], RETREE creates a three way split, where the three branches extend from one node.

An alternative approach to going through the menus of RETREE is to place the commands in an input file. For example, a file named "input" can contain:

```
Y
filename.ph
W
R
X
```

This "input" file tells RETREE to use the default parameters to create a tree. A tree is then generated using the sequences in filename.ph. Next, it commands the program to write out the rooted tree to a file, "outtree." The user can also write out the unrooted tree by replacing "R" with "U" in the "input" file. Once the "input" file is created, the user can enter "retree <input >screenout" at the command line. The program will run in the background, reading input from "input" and placing all the screen output in "screenout." The "outtree" file will also be created.

Implementation Overview

I. TreeMain - This class contains the main program. When called, the user must also enter a filename and a number 1, 2, or 3. The filename is used as the name for all the files generated when creating the phylogenetic tree. The number is used to specify what criteria to use when isolating homologous sections.

II. Alignment class:
1. callClustal(String, String, int) - The .clu file, .aln file, and total number of sequences are passed into this function. It calls Clustal W to do a complete multiple alignment on the .clu file. The .aln and .dnd files are created.
2. putSymbols(String, int) - This method takes in the .aln file and the total number of sequences. Exclamation points are placed in the .aln file where there is high indication of homology.

III. GroupMatch class:
1. putTogether(String, String, int) - This method takes in the .aln, .ext files and the total number of sequences. A .ext file is created which contains each sequence on one line.
2. isolate(String, int, String, int) - The .ext file, total number of sequences, the .tru file, and an integer 1-3 are passed into this method. The sections of the sequence that show homology are conserved and placed in the .tru file.

IV. MakeTree class:
1. makeCTree(String) - This method takes in the .tru file. Clustal W is called to create a tree. The unrooted tree is saved in a .ph file.
2. makePTree(String, String, String) - This method takes in the .ph file, an input file, and an output file. This method is supposed to call RETREE to create a visual display of the tree and save it in the output file. However, this function currently does not function correctly and has been commented out. RETREE must be called manually instead.

Conclusion

By automating the steps to go from amino acid sequences to phylogenetic trees, biologists and others can save time and effort. Editing, formatting, aligning, and tree building is no longer necessary to complete by hand. However this is just one of the first steps to the ultimate goal. In the future, a web site will be available to retrieve information on any gene, inform users of new genes, manipulate and analyze sequences, and create databases for new gene families.

Acknowledgements

I would like to thank all the people involved with the CRPC program. Special thanks to Svjetlana Miocinovic for being a great partner/mentor. Also, thanks to Dr. George Gutman for all the useful ideas and explanations. Thank you to Matt Ashton and Joe Kiniry for the UNIX help, Mika Nyström for helping me with the command lines, Amber Van Wyk for all the Java assistance, and JoAnn Boyd for being a magnificent coordinator of the CRPC program.

Bibliography

1. Arnold, Ken and Goslong, James. *The Java Programming Language Second Edition* . Reading, Massachusetts: Addison-Wesley Longman, Inc, 1998.

2. Chandy, K. George and Gutman, George. "Voltage-Gated Potassium Channel Genes." In *Handbook of Receptors and Channels Ligand and Voltage-Gated Ion Channels* . North, R. Alan. Boca Raton: CRC Press, 1995.

3. *ClustalW & ClustalX at CSC* . Available: http://www.csc.fi/molbio/progs/clustalw/clustalw.html. [1999, June 23].

4. Felsenstein, Joseph. *PHYLIP Home Page* . Department of Genetics at University of Washington. Available: http://evolution.genetics.washington.edu/phylip/phylip.html. [1999, August 3].

5. Felsenstein, Joseph. *The Newick Tree Format* . Department of Genetics at University of Washington. Available: http://evolution.genetics.washington.edu/phylip/newicktree.html. [1999, August 4].

6. Felsenstein, Joseph. *RETREE -- Interactive Tree Rearrangement* . Washington: University of Washington. 1993.

7. Higgins, Desmond. *Clustal V Multiple Sequence Alignments. Documentation (Installation and Usage)* . European Bioinformatics Institute. Available: http://www.csc.fi/molbio/progs/clustalw/doc.html. [1999, June 24].

8. Jackson, Jerry and McClellan Alan. *Java by Example* . Mountain View, CA: SunSoft Press, 1996.

9. *Java.sun.com - The Source for Java (TM) Technology* . Available: http://www.java.sun.com. [1999, June 21].

10. Lemay, Laura and Perkins, Charles. *Teach Yourself Java in 21 Days* . Indianapolis, IN: Sams.net Publishing, 1996.

11. Maciukenas, Mike. *TreeTool User Manual* . Available: http://www.hgmp.mrc.ac.uk/Menu/Help/treetool.html. [1999, August 17].

12. Strong, Michael and Chandy, K. George and Gutman, George A. *Molecular Evolution of Voltage-Sensitive Ion Channel Genes: On the Origins of Electrical Excitablility* . University of Chicago, 1993.

13. Swofford, David. *PAUP* 4.0* . Available: http://www.lms.si.edu/PAUP/paupfaq/index.html. [1999, June 23].

14. *The National Center for Biotechnology Information* . Available: http://www.ncbi.nlm.nih.gov. [1999, June 18].

15. Thompson, Julie and Higgins, Desmond and Gibson, Toby. *ClustalW: improving the sesitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties, and weight matrix choice* . Germany, Heidelberg.

Back to main page